Sandcastle -- tên nhóm

Architectural Drivers Specification 1.0

ACDM Architecture Tool -- tên dự án

Ahn, Cho, Chotemateepirom, Haider, Zgraggen - tên thành viên
10/10/2008 - update

Revision History

Date	Version	Author	Description

Contents

1 Introduction - update

1.1 Purpose

This document will be used to record, communicate and refine the architectural drivers for the project. This document will act as the main repository of requirements for the length of the project.

The intended audience for this document is the Sandcastle team and the customer. These stakeholders are to review the document and changes will be incorporated as per the change management process of the Sandcastle team. [SOW08]

1.2 Definitions, Acronyms and Abbreviations

2 Project Overview - update

(mô tả dự án vé xe bus online)

The ACDM Architecture Tool is a tool that will allow the software architect to draw diagrams of design of the software to be developed. Detailed project goals and all relevant contextual information for this document is presented in the [SOW08] document.

3 Architectural Drivers Overview

(giải thích ý nghĩa tài liệu architecture driver)

The architectural drivers presented in this document includett

- **Functional Requirements:** These requirements are presented in the form of specifications and use cases. These are a refinement of the requirements documented in the raw requirements specification [RS08] document.
- **Quality Attribute Requirements:** These requirements are presented in the form of quality attribute scenarios. These scenarios are based on the quality attributes documented in the raw requirements specification [RS08] document.
- **Business Constraints:** These are the business constraints documented in the raw requirements specification [RS08] document.
- **Technical Constraints:** These are the technical constraints documented in the raw requirements specification [RS08] document.

These architectural drivers will influence the architectural design and implementation of the project. Additionally, they will impact the schedule and quality of the project. As a whole these architectural drivers define the scope of the project.

The elicitation process by which these architectural drivers were gathered is detailed in the [PDP08] document.

4 Functional Requirements

4.1 Template

Title of functional requirement specification	**ID:** unique identifier.
	Priority: priority integer based on the priority table.
	Version: integer version number.
	Last Changed: Date this specification was last changed. ftoth Version and this field will be changed at the same time. Only changes that affect the functionality should affect these two fields, and not minor grammatical changes.
Description of the requirement specification.	
Open Issues: These are any ambiguities or conflicts in requirements that need to be clarified with the client. Additionally, these can be issues that the team needs to solve on their own.	

4.2 Specifications

Prioritization scale has been defined in section 7.1.

4.2.1 Allow designers to capture, categorize, and refine architectural drivers	**ID:** FR1
	Priority: 1
	Version: 1
	Last Changed: 11/22/2008
Architecture drivers should be categorized in the following categoriestt • Functional Requirement • Quality Attribute Scenarios	

- Technical Constraint
- ftusiness Constraint

Users cannot define their own categories.

Information to be captured for each architecture driver is listed in [lat08] (stage 2). Functional requirements should use the use case and entity templates. Quality attribute scenarios should use the 6 step template *([lat08] page 222, chapter 9, Table9.16)*. Simple templates for business and technical constraints are also present in [lat08].

The architect should be prompted to enter the architectural drivers before they create the design. This should be done in an unobtrusive way. Also, for each architecture driver, a certain list of actions should be performed (writing prose description, rationale). If these actions are not performed, the user should be notified, again in an unobtrusive way.

Refining architectural drivers will be facilitated in allowing annotations on elements. Architecture drivers will be mapped onto elements or relationships, and thus there will be a collection of annotations attached with a driver. (see requirements traceability table for related requirements)

The purpose of this feature is to have users use the tool as the primary storage for their architectural drivers. Thus the interface and features for this requirement must allow them to perform the basic tasks of adding and editing data and also take into account other features that they might need.

More detail about usage of architecture drivers in FR6.

Open Issues

4.2.2 Provide a drawing palette for rendering structures in views in the static, dynamic, and physical perspective that provide a collection of predefined elements and relationship types that are consistent with respect to perspective

ID: FR2
Priority: 1
Version: 1
Last Changed: 11/22/2008

List of element and relationship types to have in predefined list for each perspective (see section 8 for more information)tt

- Dynamic
  - Process
  - Threads
  - Objects
  - Data store
  - Data flow
  - ttivents
  - Interrupts
  - Call return
- Static
  - Modules
  - Procedure
  - Class
  - Uses
  - Inherits
- Physical

- Anything in the physical world

The tool will provide the user the ability to create multiple views in a particular perspective that will act as canvases for architectural diagrams.

In each view, an element or relationship must have a unique name. (see FR3 for more information)

Open Issues

4.2.3 Allow designers to define their own elements and relationship types

ID: FR3
Priority: 1
Version: 1.1
Last Changed: 1/28/2009

ttiach perspective will have its own set of element and relationship types. Across perspectives, the name of an element and relationship type must be unique. There will be no check on the actual way the element or relationship looks.

When creating a custom element or relationship type, the following information should be askedtt

- Ask whether it is an element or relationship
- Get a name
- Get a description
- Restricted to which perspective.
  - Search for the same name in all perspectives and warn the user. Don't allow them to create a duplicate. No duplicates across perspectives.
- Drawing of the element type. In case of a relationship type, it would just be properties of how the line and arrowheads look.

Users should be able to take a predefined element or relationship type, modify it and save it as a custom type. There should also be a quick way to revert back to default element and relationship types.

The user should be able to specify the color and attributes for different parts of the element and relationship types.

All shapes should be portable and should be packaged within a file if it uses custom shapes. When custom shapes are created, the default behavior should be to have those custom shapes only visible at that project level.

Open Issues
There should be an option to promote a custom shape to the application level, to make it available to other projects.

4.2.4 Provide a way to create element and relationship catalogs

ID: FR4
Priority: 1
Version: 2.1
Last Changed: 4/7/2009

The element and relationship catalogs will be at the view level. It will be a table containing all elements and relationships in the view, with the following information provided for eachtt

- ttilementtt "Responsibilities", "Provides", and "Requires"
- Relationshiptt "Responsibilities"

- Grouptt "Responsibilities".

The user should be able to enter these values (as plain text) directly into the relationship catalog.

Open Issues

4.2.5 Allow designers to capture design rationale

ID: FR5
Priority: 1
Version: 1
Last Changed: 11/22/2008

Rationale is attached with each view of a perspective. Rationale will be in the form of prose.

Open Issues

4.2.6 Allow designers to explicitly assign architectural drivers to design decisions

ID: FR6
Priority: 1
Version: 1
Last Changed: 11/22/2008

The user must be allowed to manually assign an architectural driver to particular elements and relationships in the architecture.

The architecture drivers will be used in the tool in the following waystt

1. Assign to elements or relationships
2. Assign to a decomposition mapping (between an element and a view)
3. Assign to a mapping between elements or groups of elements (FR8 & FR9)

The assignment must propagate down to child elements and relationships when an element is decomposed. Decomposing an element will result in a view. The decomposed element and the resulting view will have a parent child relationship.

Architectural drivers cannot be assigned to views.

Open Issues

4.2.7 Provide a way for designers to define element “interfaces” statically, dynamically, and physically

ID: FR7
Priority: 3
Version: 1
Last Changed: 11/22/2008

Interfaces are at the code level. This is a tentative requirement.

Open Issues

4.2.8 Allow designers to map elements to elements within a perspective

ID: FR8
Priority: 1
Version: 1
Last Changed: 11/22/2008

The user must be allowed to group elements together, and create mappings between elements or groups of elements in the same perspective.
Additionally, the user should be allowed to create mappings between views.

The mappings type can be hierarchical or non-hierarchical. They should be able to describe the mappings using prose.
Open Issues:

4.2.9 Allow designers to map elements to elements across perspectives	**ID:** FR9
	Priority: 2
	Version: 1
	Last Changed: 11/22/2008
The user must be allowed to perform the same operations as in FR8 for elements and groups of elements and views in different perspectives.	
Open Issues	

4.2.10 Provide the ability to use UML as a vocabulary for detailed design drawings	**ID:** FR10
	Priority: 3
	Version: 1
	Last Changed: 11/22/2008
Not required at the moment.	
Open Issues	

4.2.11 Provide the ability to warn users when various design obligations have not been met	**ID:** FR11
	Priority: 2
	Version: 2
	Last Changed: 4/7/2009
The user must be warned in the following cases (See section 9 for details)tt • Not assigning architecture drivers to elements and relationships. • Not having a rationale for a view. • Not having element relationship catalogues. The user needs to enter the details for catalogues. • There's a missing perspective. The above warnings could be shown as 'compiler warnings' as an example. The user can customize their own design obligations. They should be allowed to manage a todo list in which they can add task titles and priorities. They should have the ability to mark a task as completed and be able to delete tasks.	
Open Issues	

4.2.12 Allow designers to save, load, delete, and rename projects	**ID:** FR12
	Priority: 1
	Version: 1.1
	Last Changed: 1/28/2009
• Have only one project open in the application at a time.	
Open Issues • Custom shapes should be bundled with a file and imported by the tool on a different machine.	

4.2.13 Allow designers to edit projects after they have been saved	**ID:** FR13
	Priority: 1
	Version: 1
	Last Changed: 11/22/2008
Open Issues	

4.2.14 Facilitate collaborative design by allowing a community of designers to use the tool to define and update the design	**ID:** FR14
	Priority: 3
	Version: 1
	Last Changed: 11/22/2008
Not required at the moment.	
Open Issues	

4.2.15 All interaction between designers and the tool will be through a graphical user interface	**ID:** FR15
	Priority: 1
	Version: 1
	Last Changed: 11/22/2008
Open Issues	

4.2.16 Print (or Presentation mode) formatting requirements	**ID:** FR16
	Priority: 3
	Version: 1
	Last Changed: 11/22/2008
Format and convert the contents of a file to a Microsoft Word document so the user can manually format the document and create a print ready version.	
Open Issues • We need to design this in more detail. We can discuss our design with the client when we have it.	

4.3 Specifications Traceability Matrix --- (optional)

This traceability matrix identifies dependencies between functional requirements. Dependencies are identified based on explicit dependencies mentioned in the text of the functional requirement specifications and implicit dependencies that are implied by the functions being documented. tti.g. consider the dependency between FR1 and FR2, which exists because architectural drivers captured in FR1 are mapped to elements created in FR2 (in this case, the dependency originates from FR6).

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10	FR11	FR12	FR13	FR14	FR15	FR16
FR1		X				X					X	X	X		X	X
FR2	X		X	X		X						X	X		X	X
FR3		X				X						X	X		X	
FR4		X									X	X	X		X	X
FR5		X									X	X	X		X	X
FR6	X	X	X					X	X		X	X	X		X	X
FR7																
FR8						X			X			X	X		X	X
FR9						X		X				X	X		X	X
FR10																
FR11	X			X	X	X									X	X
FR12	X	X	X	X	X	X		X	X				X		X	
FR13	X	X	X	X	X	X		X	X			X			X	
FR14																
FR15	X	X	X	X	X	X		X	X		X	X	X			X
FR16	X	X		X	X	X		X	X		X				X	

4.4 Use Case Modeling

4.4.1 Domain Model

Domain Object	Description
Use case	A collection of information that the end user will enter into the tool. This collection will includett Title, ID, Description, ttintities Involved, Preconditions, Primary use case flow of events, Primary use case postconditions, Alternate user case flow of events, Alternate use case postconditions.
Entity	A collection of information that the end user will enter into the tool. This collection will includett Name, ID, Description, Provides Assumptions, Requires Assumptions, Identified user cases.
Quality attribute	A collection of information that the end user will enter into the tool. This collection will includett Title of scenario, ID, Quality attribute, Description of stakeholder's role, Source of the stimulus, Stimulus, Relevant environmental conditions, Architectural elements, System response, and Response measure.
Business constraint	A collection of information that the end user will enter into the tool. This collection will includett ID, Title, Consideration.
Technical constraints	A collection of information that the end user will enter into the tool. This collection will includett ID, Title, Consideration.
Element type	A collection of information that the end user will enter into the tool. This collection will includett Type, Name, Description, Perspective.
Relationship type	A collection of information that the end user will enter into the tool. This collection will includett Type, Name, Description, Perspective.

4.4.2 Entities

Entity name: Architect	**Entity ID: E01**
Description:	

Architects are people who use the tool to design their software intensive system they want to build. They are the only end users of the tool. Architects are assumed to be familiar with ACDM, adept at using computers, and responsible for creating and maintaining architectures based on the ACDM methodology.
Provides assumptions: Architects have gathered the architectural drivers and finished ACDM stage 2.
Requires assumptions: Architects expect no interruptions from the tool. He wants the freedom to draw the design in various ways. The only feedback the Architects expect from the tool are warnings if the design is not consistent and regulations from the ACDM methodology are not met.
Identified use cases: Architects are the single entity for this tool. Therefore, all the use cases are identified for this entity.

4.4.3 Use Case List

UC ID	UC Title	FR ID
UC01	Capture functional requirement	FR1
UC02	Capture quality attribute	FR1
UC03	Capture technical constraint	FR1
UC04	Capture business constraint	FR1
UC05	Create/Remove a view	FR2
UC06	Draw a design in a view	FR2
UC07	Define new element or relationship type	FR3
UC08	ttidit element or relationship type	FR3
UC09	Remove element or relationship type	FR3
UC10	Reset toolbox	FR3
UC11	Manage element and relationship catalogs	FR4
UC12	Capture design rationale for view	FR5
UC13	('Capture design rationale for an element, relationship or group' is removed – at version 1.6)	
UC14	Assign/Remove an architectural driver to/from an element/relationship/ group	FR6
UC15	View/Remove an element or relationship to which an architectural driver is assigned	FR6
UC16	Decompose an element/relationship/group	FR6, FR8
UC17	Create a mapping between different views	FR8
UC18	Create a mapping for element/relationship/group	FR8
UC19	Delete a mapping	FR8
UC20	Capture design rationale for a mapping	FR5, FR8
UC21	Assign/Remove architectural drivers to/from a mapping of elements	FR8, FR9
UC22	Use UML to draw detail design	FR10
UC23	Warn user about unmet design obligations	FR11
UC24	(silence warning is removed – at version 1.2)	
UC25	Manage tasks for the project	FR11
UC26	Manage a project	FR3, FR12
UC27	ttixport a project	FR16
UC28	Group elements and relationships	FR8
UC29	Ungroup elements and relationships	FR8
UC30	View decomposition of an element	FR6, FR8
UC31	ttixport an architecture project to a document format	FR16

4.4.4 Use Cases

4.4.4.1 Capture functional requirement

Use case title: Capture functional requirements			**Use case ID:** UC01 **Version:** 1 **Last Changed:** 11/22/2008
General use case description: **Architect will add/edit/delete functional requirement.**			
Entities involved: **E01 – Architect**			
Preconditions: **None.**			
Primary use case flow of events:			
1.	tti01 selects the architectural drivers view.		
2.	tti01 selects the functional requirement view.		
3	**If** tti01 wants to add a functional requirement,		
	1.	tti01 enters a use case.	
4.	**If** tti01 wants to delete a functional requirement,		
	1.	tti01 selects a use case.	
	2.	tti01 deletes the selected use case.	
5.	**If** tti01 wants to edit a functional requirement,		
	1.	tti01 selects a use case.	
	2.	tti01 edits the selected use case.	
Alternate use case 3.1 flow of events:			
1.	tti01 can add a new entity for the use case		
2.	tti01 enters an entity.		
Alternate use case 4.1 flow of events:			
1.	tti01 can delete an entity for the use case		
2.	tti01 selects the entity.		
3.	tti01 deletes the entity.		
Alternate use case 4.2 flow of events:			
1.	If the functional requirement is assigned to something in the drawing. tti01 will be asked to confirm it if he still wants to proceed.		
2.	The system will delete all related assignments.		
Alternate use case 5.2 flow of events:			
1.	tti01 can edit an entity for the use case		
2.	tti01 selects the entity.		
3.	tti01 edits the entity.		
Primary use case postconditions: **The functional requirement is saved in the system.**			

4.4.4.2 Capture quality attribute

Use case title: Capture quality attribute			**Use case ID:** UC02 **Version:** 1 **Last Changed:** 11/22/2008
General use case description: **Architect will add/edit/delete a quality attribute.**			

Entities involved: **E01 – Architect**		
Preconditions: **None.**		
Primary use case flow of events:		
1.	tti01 selects the architectural drivers view.	
2.	tti01 selects the quality attributes view.	
3.	**If** tti01 wants to add a quality attribute,	
	1.	tti01 enters a quality attribute.
4.	**If** tti01 wants to remove a quality attribute,	
	1.	tti01 selects a quality attribute.
	2.	tti01 deletes the quality attribute.
5.	**If** tti01 wants to edit a quality attribute,	
	1.	tti01 selects a quality attribute.
	2.	tti01 edits the quality attribute.
Alternate use case 4.2 flow of events:		
1.	If the quality attribute is assigned to something in the drawing. tti01 will be asked to confirm it if he still wants to proceed.	
2.	The system will delete all related assignments.	
Primary use case postconditions: **The quality attribute is saved in the system.**		

4.4.4.3 Capture technical constraint

Use case title: Capture technical constraint	**Use case ID:** UC03 **Version:** 1 **Last Changed:** 11/22/2008	
General use case description: **Architect will add/edit/delete a technical constraint**		
Entities involved: **E01 – Architect**		
Preconditions: **None.**		
Primary use case flow of events:		
1.	tti01 selects the architectural drivers view.	
2.	tti01 selects the technical constraints view.	
3.	**If** tti01 wants to add a technical constraint,	
	1.	tti01 enters a technical constraint.
4.	**If** tti01 wants to remove a technical constraint,	
	1.	tti01 selects a technical constraint.
	2.	tti01 deletes the technical constraint.
5.	**If** tti01 wants to edit a technical constraint,	
	1.	tti01 selects a technical constraint.
	2.	tti01 edits the technical constraint.
Alternate use case 4.2 flow of events:		
1.	If the technical constraint is assigned to something in the drawing. tti01 will be asked to confirm it if he still wants to proceed.	
2.	The system will delete all related assignments.	
Primary use case postconditions:		

The technical constraint is saved in the system.

4.4.4.4 Capture business constraint

Use case title: Capture business constraint		**Use case ID:** UC04
		Version: 1
		Last Changed: 11/22/2008
General use case description: **Architect will add/edit/delete a business constraint**		
Entities involved: **E01 – Architect**		
Preconditions: **None.**		
Primary use case flow of events:		
1.	tti01 selects the architectural drivers view.	
2.	tti01 selects the business constraints view.	
3.	**If** tti01 wants to add a business constraint,	
	1. tti01 enters a business constraint.	
4.	**If** tti01 wants to remove a business constraint,	
	1. tti01 selects a business constraint.	
	2. tti01 deletes the business constraint.	
5.	**If** tti01 wants to edit a business constraint,	
	1. tti01 selects a business constraint.	
	2. tti01 edits the business constraint.	
Alternate use case 4.2 flow of events:		
1.	If the business constraint is assigned to something in the drawing. tti01 will be asked to confirm it if he still wants to proceed.	
2.	The system will delete all related assignments.	
Primary use case postconditions: **The business constraint is saved in the system.**		

4.4.4.5 Create/Remove a view

Use case title: Create/Remove a view		**Use case ID:** UC05
		Version: 1
		Last Changed: 11/22/2008
General use case description: **Architect can create/remove a view from perspectives.**		
Entities involved: **E01 – Architect**		
Preconditions: **None.**		
Primary use case flow of events:		
1.	tti01 selects a perspective.	
2.	**If** tti01 wants to create new view,	
	1. tti01 selects the option to create view.	
	2. tti01 enters name for the view.	
	3. The system creates the view.	
3.	**If** tti01 wants to remove a view,	
	1. tti01 selects a view from the list.	
	2. tti01 selects the option to remove the view.	

	3.	tti01 confirms removing the view.
Primary use case postconditions: **The view is created or removed to the selected perspective.**		
Alternate use case 2.2 flow of events:		
1.	If the name that tti01 entered is already exist in the project, tti01 needs to choose another name.	
Alternate use case 2.3 flow of events:		
1.	If the view is mapped to another view, tti01 will be asked if he wants to proceed and delete the view and its mappings.	
2.	The system deletes the view, its mappings and all child elements, relationships and their assignments, mappings and child views (or decomposed elements).	

4.4.4.6 Draw a design in a view

Use case title: Draw a design in a view		**Use case ID:** UC06
		Version: 3
		Last Changed: 12/3/2008
General use case description: **Architect can draw diagram to the view.**		
Entities involved: **E01 – Architect**		
Preconditions: **A view exists and is currently being viewed.**		
Primary use case flow of events:		
1.	tti01 selects a perspective.	
2.	tti01 selects a view.	
3.	**If** tti01 wants to add an element or relationship,	
	1.	tti01 selects an element or relationship from the toolbox of the current perspective.
	2.	tti01 adds the selected element or relationship to the view.
	3.	tti01 names element or relationship.
4.	**If** tti01 wants to remove an element or relationship,	
	1.	tti01 selects an element or relationship in the view.
	2.	tti01 select the option to remove the element or relationship from the view.
	3.	The system deletes all assignments, and mappings for the element or relationship.
5.	**If** tti01 wants to edit name of element or relationship,	
	1.	tti01 selects an element or relationship in the view.
	2.	tti01 renames the element or relationship.
6.	The system updates the diagram.	
Primary use case postconditions: **The changes are reflected on the diagram.**		
Alternate use case 3.3 flow of events:		
1.	tti01 does not have to specify a name for an element or relationship.	
Alternate use case 3.3 and 5.2 flow of events:		
1.	If an element with the name that tti01 entered already exists in the same view, tti01 needs to choose another name.	
Alternate use case 4.3 flow of events:		
1.	If the element or relationship has a mapping to another element or group, or if the element has an assigned architectural drivers, the tti01 will be asked if he wants to proceed and delete the element or relationship and delete the mapping or assignment	
Alternate use case 4.3 flow of events:		
1.	If the element has a mapping to a decomposed view tti01 will be given the choice of removing just the	

	element, or the element and its child view together.
Note: An Element and a Relationship can't have the same name.	

4.4.4.7 Define new element or relationship type

Use case title: Define new element or relationship type			**Use case ID:** UC07
			Version: 1.1
			Last Changed: 1/28/2009
General use case description: **An architect creates new element and relationship type.**			
Entities involved: **E01 – Architect**			
Preconditions: **None.**			
Primary use case flow of events:			
1.	tti01 selects a perspective.		
2.	**If** tti01 selects the option to add a new element type,		
	1.	tti01 specifies values for new element type.	
	2.	tti01 draws the new element type in the drawing area. The drawing area includes basic shapes and supports the change of the line type (full, dots, etc), and the drawing color.	
3.	**If** tti01 selects the option to add a new relationship type,		
	1.	tti01 specifies values for new relationship type.	
	2.	tti01 specifies properties of the line (full, dots, etc), and the line color.	
Primary use case postconditions: **The changes are reflected on the element and relationship type list. This new type will be visible only in project level, not in application level.**			
Alternate use case 2.1 and 3.1 flow of events:			
1.	If the name that tti01 entered already exists in the element and relationship type list in the same perspective, tti01 needs to choose another name.		
Alternate use case 2 and 3 flow of events:			
1.	tti01 can choose to use an existing element or relationship type to create a new element or relationship.		
2.	The system will then use the existing element or relationship type as a base. ftut only the shape is used. The name, description etc. are removed and have to be entered by the user.		
Open Issues			

4.4.4.8 Edit element or relationship type

Use case title: Edit element or relationship type			**Use case ID:** UC08
			Version: 1.1
			Last Changed: 1/28/2009
General use case description: **An architect edits an element or relationship type.**			
Entities involved: **E01 – Architect**			
Preconditions: **An element or relationship type exists.**			
Primary use case flow of events:			
1.	tti01 selects a perspective.		
2.	tti01 selects the element or relationship type.		
3.	tti01 selects the option to edit this element or relationship type.		

4.	tti01 changes the values for the element or relationship type (Type, Name, Description and look and feel).
5.	tti01 makes changes to the element or relationship type in the drawing area.
6.	The system updates the element or relationship both on the toolbox and on the canvas.
Primary use case postconditions: **The changes are reflected on the element and relationship type toolbox. The change will affect project level, not application level.**	
Alternate use case 4 flow of events:	
1.	If the name that tti01 entered already exists in the element and relationship list in the same perspective, tti01 needs to choose another name.
Open Issues	

4.4.4.9 Remove element or relationship type

Use case title: Remove element or relationship type	**Use case ID:** UC09 **Version:** 1.1 **Last Changed:** 1/28/2009
General use case description: **An architect removes an element or relationship type.**	
Entities involved: **E01 – Architect**	
Preconditions: **An element or relationship type exists.**	
Primary use case flow of events:	
1.	tti01 selects a perspective.
2.	tti01 selects the element or relationship type.
3.	tti01 selects the option to remove this element or relationship type.
4.	tti01 confirms removing the element or relationship type.
5.	System removes the element or relationship type.
Primary use case postconditions: **The changes are reflected on the element and relationship list. The change will affect project level, not application level.**	
Alternate use case 4 flow of events:	
1.	If the element or relationship type has instances in the drawing then it is not possible to delete the type. Inform tti01 accordingly.
Open Issues	

4.4.4.10 Reset toolbox

Use case title: Reset toolbox	**Use case ID:** UC10 **Version:** 1.1 **Last Changed:** 1/28/2009
General use case description: **An architect resets the toolbox. This will remove all the custom shapes the user defined.**	
Entities involved: **E01 – Architect**	
Preconditions: **None**	
Primary use case flow of events:	
1.	tti01 selects the option to reset the toolbox.
2.	tti01 confirms removing the element or relationship type.

3.	The system updates the element or relationship both on the toolbox and on the canvas.
Primary use case postconditions: **The changes are reflected on the element and relationship list. The change will affect project level, not application level.**	
Alternate use case 2 flow of events:	
1.	If some element or relationship types have instances. tti01 will be asked if he wants to delete all the other custom shapes which don't have instances.
Open Issues	

4.4.4.11 Manage element and relationship catalogs

Use case title: Manage element and relationship catalogs	**Use case ID:** UC11 **Version:** 2.1 **Last Changed:** 4/7/2009
General use case description: **Architects can provide information for each element, relationship, or group in the view.**	
Entities involved: **E01 – Architect**	
Preconditions: **An element or relationship is already created.**	
Primary use case flow of events:	
1.	tti01 selects a perspective and view.
2.	tti01 selects the option to manage catalogs.
3.	The system shows the catalog of the view.
4.	**If** tti01 selects an element,
	1. tti01 edits information or enters new information. Information includes "Responsibilities", "Requires" and "Provides".
5.	**If** tti01 selects a relationship or group,
	1. tti01 edits information or enters new information. Information includes "Responsibilities".
Primary use case postconditions: **Information is updated to the catalogs.**	

4.4.4.12 Capture design rationale for view

Use case title: Capture design rationale for view	**Use case ID:** UC12 **Version:** 1 **Last Changed:** 11/22/2008
General use case description: **Rationale is attached to each view of a perspective. Rationale will be in the form of prose.**	
Entities involved: **E01 - Architect**	
Preconditions: **A view is already created.**	
Primary use case flow of events:	
1.	tti01 selects the view and selects the option to capture rationale.
2.	The system shows the current rationale of the selected view.
3.	tti01 edits rationale or enters new rationale.
Primary use case postconditions: **The rationale is saved to the selected view.**	

~~4.4.4.13 Capture design rationale for an element, relationship or group~~

4.4.4.14 Assign/Remove an architectural driver to/from an element/relationship/group of elements and relationships

Use case title: Assign/Remove an architectural driver to/from an element/relationship/group		**Use case ID:** UC14 **Version:** 3 **Last Changed:** 12/3/2008
General use case description: **Architects can assign and unassign an architectural driver to particular elements and relationships or groups of them in the architecture.** **Architectural drivers cannot be assigned to views.**		
Entities involved: **E01 – Architect**		
Preconditions: **An element or relationship is already created. At least one element or relationship or a group exists.**		
Primary use case flow of events:		
1.	tti01 selects an element or relationship or a group.	
2.	**If** tti01 wants to assign an architectural driver to the selected element or relationship or group,	
	1.	tti01 selects the option to assign an architectural driver.
	2.	The system shows the list of architectural drivers of the project.
	3.	tti01 selects an architectural driver from the list.
	4.	The system assigns the selected architectural driver to the element or relationship or group.
3.	**If** tti01 wants to remove an assigned architectural driver from the selected element or relationship or group,	
	1.	tti01 selects the option to remove an architectural driver.
	2.	The system shows the list of architectural drivers assigned to the element or relationship.
	3.	tti01 selects the option to remove architectural driver that they want to remove.
	4.	tti01 confirms removing the selected architectural driver.
	5.	The system removes the assignment between the element or relationship or group and the selected architectural driver.
Primary use case postconditions: **The selected architectural driver is assigned/removed to/from the selected element or relationship or group.**		
Alternate use case 2.3 flow of events:		
1.	The system shows a warning because the architectural driver is already assigned to this element or relationship or group.	
Open Issues		

4.4.4.15 View/Remove an element or relationship to which an architectural driver is assigned

Use case title: View/Remove an element or relationship to which an architectural driver is assigned		**Use case ID:** UC15 **Version:** 2 **Last Changed:** 12/3/2008
General use case description: **Architects can view elements or relationships assigned to architectural drivers in the project.**		
Entities involved: **E01 – Architect**		
Preconditions: **Architectural drivers are already created. At least one element or relationship exists.**		
Primary use case flow of events:		
1.	tti01 selects an architectural driver.	
2.	The system shows a list of all elements and relationships assigned to that particular architectural driver.	